MORSE CODE TRANSLATOR

```
#include <iostream>
#include <string>
#include <map>

using namespace std;

map<char, string> morseCodeMap = {
    {'A', ".-"}, {'B', "-..."}, {'C', "-.-."},
    {'D', "-.."}, {'E', "."}, {'F', "..-."},
    {'G', "--."}, {'H', "...."}, {'I', ".."},
    {'J', ".---"}, {'K', "-.-"}, {'L', ".-.."},
    {'M', "--"}, {'N', "-."}, {'O', "---"},
    {'P', ".--."}, {'Q', "--.-"}, {'R', ".-."},
    {'S', "..."}, {'T', "-"}, {'U', "..-"},
    {'V', "...-"}, {'W', ".--"}, {'X', "-..-"},
    {'Y', "-.--"}, {'Z', "--.."},
    {'a', ".-"}, {'b', "-..."}, {'c', "-.-."},
    {'d', "-.."}, {'e', "."}, {'f', "..-."},
    {'g', "--."}, {'h', "...."}, {'i', ".."},
    {'j', ".---"}, {'k', "-.-"}, {'l', ".-.."},
    {'m', "--"}, {'n', "-."}, {'o', "---"},
    {'p', ".--."}, {'q', "--.-"}, {'r', ".-."},
    {'s', "..."}, {'t', "-"}, {'u', "..-"},
    {'v', "...-"}, {'w', ".--"}, {'x', "-..-"},
    {'y', "-.--"}, {'z',"--.."},
    {'0', "-----"}, {'1', ".----"}, {'2', "..---"},
    {'3', "...--"}, {'4', "....-"}, {'5', "....."},
    {'6', "-...."}, {'7', "--..."}, {'8', "---.."},
    {'9', "----."},
    {' ', " "}, // Space
    {'&', ".-..."}, // Example: '&' represents "ET" (Ampersand)
    {',', "--..--"}, // Comma
    {'.', ".-.-.-"}, // Period (Full stop)
    {'?', "..--.."}, // Question Mark
    {'!', "-.-.--"},
    {'@', ".--.-."}, // At Symbol
    {'_', "..--.-"}, // Underscore
    {'\'', ".----."}, // Apostrophe
    {'"', ".-..-."}, // Quotation Mark
    {'(', "-.--."}, // Left Parenthesis
    {')', "-.--.-"}, // Right Parenthesis
    {'=', "-...-"}, // Equals Sign
    {'+', ".-.-."}, // Plus Sign
```

```
    {'-', "-....-"}, // Hyphen
    {'/', "-..-."}, // Forward Slash
    {';', "-.-.-."}, // Semicolon
    {':', "---..."}, // Colon
    {'$', "...-..-"}, // Dollar Sign
    {'%', "-..-.,"}, // Percent Sign
    {'#', ".-.-."}, // Hash Sign
    {'*', "-..-"}, // Asterisk
    {'[', "-.--."}, // Left Square Bracket
    {']', "-.--.-"}, // Right Square Bracket
    {'{', ".--.-"}, // Left Curly Brace
    {'}', ".--.-."}, // Right Curly Brace
    {'<', "-.-.--"}, // Less Than Sign
    {'>', "-.-.-."}, // Greater Than Sign
    {'|', "-...-"},// Exclamation Mark
};

string textToMorseCode(const string& text) {
    string morseCode;
    for (char ch : text) {
        if (isalpha(ch)) {
            ch = toupper(ch);
        }
        morseCode += morseCodeMap[ch] + " ";
    }
    return morseCode;
}

string morseCodeToText(const string& morseCode) {
    string text;
    string currentSymbol;

    for (char ch : morseCode) {
        if (ch == ' ' && !currentSymbol.empty()) {
            for (const auto& pair : morseCodeMap) {
                if (pair.second == currentSymbol) {
                    text += pair.first;
                    break;
                }
            }
            currentSymbol.clear();
        } else if (ch != ' ') {
            currentSymbol += ch;
        }
```

```
    }

    if (!currentSymbol.empty()) {
        for (const auto &pair : morseCodeMap) {
            if (pair.second == currentSymbol) {
                text += pair.first;
                break;
            }
        }
    }

    return text;
}

int main() {
    cout << "Morse Code Translator" << endl;
    cout << "Enter '1' to convert text to Morse code, '2' to convert Morse code to text: ";
    char choice;
    cin >> choice;

   cin.ignore(); // Ignore any remaining characters in the input buffer, including the newline
character.

    if (choice == '1') {
        cout << "Enter the text to convert to Morse code: ";
        string inputText;
        getline(cin, inputText);
        string morseCode = textToMorseCode(inputText);
        cout << "Morse code: " << morseCode << endl;
    } else if (choice == '2') {
        cout << "Enter the Morse code to convert to text: ";
        string inputMorseCode;
        getline(cin, inputMorseCode);
        string text = morseCodeToText(inputMorseCode);
        cout << "Text: " << text << endl;
    } else {
        cout << "Invalid choice. Exiting the program." << endl;
        return 1; // Return an error code to indicate an issue occurred.
    }

    return 0;
}
```